In [ ]:
```
import pandas as pd
import numpy as np
```

In [ ]:
```
df = pd.read_csv('data/titanic.csv')
df
```

Out[ ]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

In [ ]:
```
df.describe()
```

Out[ ]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [ ]:
```
df.nunique()
```

Out[ ]:
```
PassengerId    891
Survived         2
Pclass           3
Name           891
Sex              2
Age             88
SibSp            7
Parch            7
Ticket         681
Fare           248
Cabin          147
Embarked         3
dtype: int64
```

In [ ]:
```
df.duplicated().sum()
```

Out[ ]:
```
0
```

In [ ]:
```
df.isna().sum()
```

Out[ ]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

Cabin column is dropped because of too many missing values Name and Ticket dropped because it is unique at each column

In [ ]:
```
df2 = df.drop(columns=['Name','Ticket','Cabin'])
```

Dropped 2 rows where embarked is null

In [ ]:
```
df2.dropna(inplace=True, subset=['Embarked'])
```

In [ ]:
```
data = df2.values
X = data[:, 2:]
print(X)
```

```
[[3 'male' 22.0 ... 0 7.25 'S']
 [1 'female' 38.0 ... 0 71.2833 'C']
 [3 'female' 26.0 ... 0 7.925 'S']
 ...
 [3 'female' nan ... 2 23.45 'S']
 [1 'male' 26.0 ... 0 30.0 'C']
 [3 'male' 32.0 ... 0 7.75 'Q']]
```

In [ ]:
```
y = data[:,1]
print(y)
```

```
[0 1 1 1 0 0 0 0 1 1 1 1 0 0 0 1 0 1 0 1 0 1 1 1 0 1 0 0 1 0 0 1 1 0 0 0 1
 0 0 1 0 0 0 1 1 0 0 1 0 0 0 0 0 1 1 0 1 1 0 1 0 0 0 0 0 1 1 0 1 0 0 0 0 0 1
 0 0 0 1 1 0 1 1 0 1 1 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1 0 1 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 0 0 0 0 1 0 0 1 0 0 0 0 1 1 0 0 0 1 0 0
 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0
 1 1 0 0 1 0 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 0 0
 0 1 0 1 0 0 0 1 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 1 1 1
 0 1 0 0 0 0 0 1 1 1 0 1 1 0 1 1 0 0 0 1 0 0 0 1 0 0 1 0 1 1 1 1 0 0 0 0 0
 0 1 1 1 1 0 1 0 1 1 1 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 1 0 1 0 1 1 1 1 0 0 0
 1 0 0 1 1 0 1 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 1 1
 0 0 0 0 1 1 0 0 0 1 1 0 1 0 0 0 1 0 1 1 1 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0 1
 0 0 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1
 1 1 1 1 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 1 0 0
 0 1 1 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 1 0 1 1 0 1 1 0 0 1 0 1
 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 1 0 1 0 1 1 0 0 1 0 0 1 1 0 1 1 0 0 1 1 0
 1 0 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 1 1 0 1 1 1 0 0 0 1 0 1 0 0 0 1 0
 0 0 0 1 0 0 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 0 1 0 0 1 0 0 1 1 0 0 0 0 1 0 0
 1 0 1 0 0 1 0 0 0 0 0 1 0 1 1 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0
 0 0 1 1 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0 0 1 1 0 0
 0 0 1 1 1 1 1 0 1 0 0 0 1 1 0 0 1 0 0 0 1 0 1 1 0 0 1 0 0 0 0 0 0 1 0 0 1
 0 1 0 1 0 0 1 0 0 1 1 0 0 1 1 0 0 0 1 0 0 1 1 0 1 0 0 0 0 0 0 0 0 1 0 0 1
 0 1 1 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 1 0 0 0 1 1 1 1 0 0 0 0 1 0 0 0 0 0
 0 0 0 0 0 0 1 1 0 1 0 0 0 1 1 1 1 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 1 0 0 0
 1 0 1 1 1 1 0 0 0 1 0 0 1 1 0 0 1 0 1 0 0 1 1 0 0 0 1 1 0 0 0 0 0 0 1 0 1
 0]
```

In [ ]:
```
from sklearn.model_selection import train_test_split
```

Splitting the array into test and train sets

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=1)
```

In [ ]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(711, 7)
(178, 7)
(711,)
(178,)
```

In [ ]:
```
print(X_train)
```

```
[[3 'male' nan ... 0 8.05 'S']
 [3 'male' 19.0 ... 0 10.1708 'S']
 [3 'male' nan ... 0 7.75 'Q']
 ...
 [3 'male' 26.0 ... 0 14.4542 'C']
 [2 'male' 44.0 ... 0 26.0 'S']
 [3 'male' 21.0 ... 0 8.05 'S']]
```

In [ ]:
```
from sklearn.impute import SimpleImputer
```

Imputer to fill null values using mode

In [ ]:
```
imputer = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
```

Replacing null values in age with mode

In [ ]:
```
imputer.fit(X_train[:, 2:3])
```

Out[ ]:
```
SimpleImputer
SimpleImputer(strategy='most_frequent')
```

In [ ]:
```
X_train[:, 2:3] = imputer.transform(X_train[:, 2:3])
print(X_train)
```

```
[[3 'male' 24.0 ... 0 8.05 'S']
 [3 'male' 19.0 ... 0 10.1708 'S']
 [3 'male' 24.0 ... 0 7.75 'Q']
 ...
 [3 'male' 26.0 ... 0 14.4542 'C']
 [2 'male' 44.0 ... 0 26.0 'S']
 [3 'male' 21.0 ... 0 8.05 'S']]
```

In [ ]:
```
X_test[:, 2:3] = imputer.transform(X_test[:, 2:3])
print(X_test)
```

```
[[2 'female' 36.0 ... 0 13.0 'S']
 [2 'female' 50.0 ... 1 26.0 'S']
 [3 'male' 48.0 ... 0 7.8542 'S']
 ...
 [1 'male' 22.0 ... 0 135.6333 'C']
 [3 'male' 22.0 ... 0 8.05 'S']
 [2 'male' 24.0 ... 0 10.5 'S']]
```

Min Max Feature Scaling on columns Age and Fare

In [ ]:
```
from sklearn.preprocessing import MinMaxScaler
min_max_scaler = MinMaxScaler(feature_range=(0,1))
```

In [ ]:
```
X_train[:,(2,5)] = min_max_scaler.fit_transform(X_train[:,(2,5)])
X_test[:,(2,5)] = min_max_scaler.transform(X_test[:,(2,5)])
```

In [ ]:
```
print(X_train)
```

```
[[3 'male' 0.2963056044232219 ... 0 0.03060836501901141 'S']
 [3 'male' 0.2334757476752953 ... 0 0.038672243346007606 'S']
 [3 'male' 0.2963056044232219 ... 0 0.029467680608365018 'Q']
 ...
 [3 'male' 0.32143754712239253 ... 0 0.05495893536121673 'C']
 [2 'male' 0.5476250314149284 ... 0 0.09885931558935361 'S']
 [3 'male' 0.2586076903744659 ... 0 0.03060836501901141 'S']]
```

In [ ]:
```
print(X_test)
```

```
[[2 'female' 0.4470972606182458 ... 0 0.049429657794676805 'S']
 [2 'female' 0.6230208595124404 ... 1 0.09885931558935361 'S']
 [3 'male' 0.5978889168132697 ... 0 0.029863878326996197 'S']
 ...
 [1 'male' 0.2711736617240512 ... 0 0.515715969581749 'C']
 [3 'male' 0.2711736617240512 ... 0 0.03060836501901141 'S']
 [2 'male' 0.2963056044232219 ... 0 0.039923954372623575 'S']]
```

In [ ]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LogisticRegression
from matplotlib import pyplot as plt
from sklearn import metrics
```

In [ ]:
```
y_train = y_train.astype('int')
y_test = y_test.astype('int')
```

In [ ]:
```
logr = LogisticRegression()
logr.fit(X_train, y_train)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/home/neeraj/Desktop/ICTAK/ICTAK-ML-AI-course/Assignments/session 11/Neeraj_Assignment_Session_11_titanic.ipynb Cell 32 in <cell line: 2>()
      <a href='vscode-notebook-cell:/home/neeraj/Desktop/ICTAK/ICTAK-ML-AI-course/Assignments/session%2011/Neeraj_Assignment_Session_11_titanic.ipynb#X54sZmlsZQ%3D%3D?line=0'>1</a> logr = LogisticRegression()
----> <a href='vscode-notebook-cell:/home/neeraj/Desktop/ICTAK/ICTAK-ML-AI-course/Assignments/session%2011/Neeraj_Assignment_Session_11_titanic.ipynb#X54sZmlsZQ%3D%3D?line=1'>2</a> logr.fit(X_train, y_train)

File ~/.local/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:1138, in LogisticRegression.fit(self, X, y, sample_weight)
   1135 else:
   1136     _dtype = [np.float64, np.float32]
-> 1138 X, y = self._validate_data(
   1139     X,
   1140     y,
   1141     accept_sparse="csr",
   1142     dtype=_dtype,
   1143     order="C",
   1144     accept_large_sparse=solver not in ["liblinear", "sag", "saga"],
   1145 )
   1146 check_classification_targets(y)
   1147 self.classes_ = np.unique(y)

File ~/.local/lib/python3.10/site-packages/sklearn/base.py:596, in BaseEstimator._validate_data(self, X, y, reset, validate_separately, **check_params)
    594         y = check_array(y, input_name="y", **check_y_params)
    595     else:
--> 596         X, y = check_X_y(X, y, **check_params)
    597     out = X, y
    599 if not no_val_X and check_params.get("ensure_2d", True):

File ~/.local/lib/python3.10/site-packages/sklearn/utils/validation.py:1074, in check_X_y(X, y, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, multi_output, ensure_min_samples, ensure_min_features, y_numeric, estimator)
   1069         estimator_name = _check_estimator_name(estimator)
   1070     raise ValueError(
   1071         f"{estimator_name} requires y to be passed, but the target y is None"
   1072     )
-> 1074 X = check_array(
   1075     X,
   1076     accept_sparse=accept_sparse,
   1077     accept_large_sparse=accept_large_sparse,
   1078     dtype=dtype,
   1079     order=order,
   1080     copy=copy,
   1081     force_all_finite=force_all_finite,
   1082     ensure_2d=ensure_2d,
   1083     allow_nd=allow_nd,
   1084     ensure_min_samples=ensure_min_samples,
   1085     ensure_min_features=ensure_min_features,
   1086     estimator=estimator,
   1087     input_name="X",
   1088 )
   1090 y = _check_y(y, multi_output=multi_output, y_numeric=y_numeric, estimator=estimator)
   1092 check_consistent_length(X, y)

File ~/.local/lib/python3.10/site-packages/sklearn/utils/validation.py:856, in check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, estimator, input_name)
    854         array = array.astype(dtype, casting="unsafe", copy=False)
    855     else:
--> 856         array = np.asarray(array, order=order, dtype=dtype)
    857 except ComplexWarning as complex_warning:
    858     raise ValueError(
    859         "Complex data not supported\n{}\n".format(array)
    860     ) from complex_warning

ValueError: could not convert string to float: 'male'
```

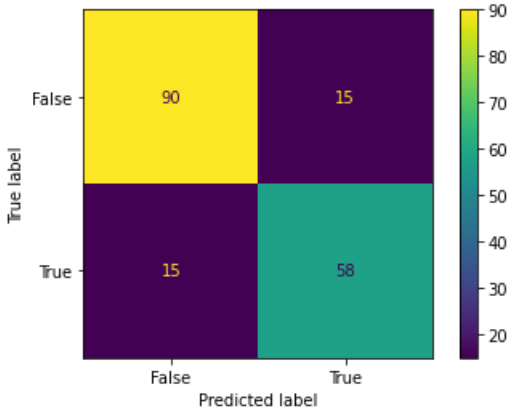
In [ ]:
```
log_pred = logr.predict(X_test)
log_pred
```

Out[ ]:
```
array([1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0,
       0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1,
       0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 0])
```

In [ ]:
```
confusion_matrix = metrics.confusion_matrix(y_test, log_pred)
confusion_matrix
```

Out[ ]:
```
array([[90, 15],
       [15, 58]])
```

In [ ]:
```
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix=confusion_matrix, display_labels=[False, True])
cm_display.plot()
plt.show()
```



In [ ]:
```
accuracy = metrics.accuracy_score(y_test, log_pred)
accuracy
```

Out[ ]:
```
0.8314606741573034
```

In [ ]:
```
precision = metrics.precision_score(y_test, log_pred)
precision
```

Out[ ]:
```
0.7945205479452054
```